For the second part of the project, we switched to Python for the path planning because it offers more useful libraries. We used the A* algorithm and allowed 18 different types of moves: the four cardinal directions (up, down, left, right), the four diagonals, pure rotation moves, and combinations of rotation plus movement. We skipped diagonal-plus-rotation to keep things simpler.

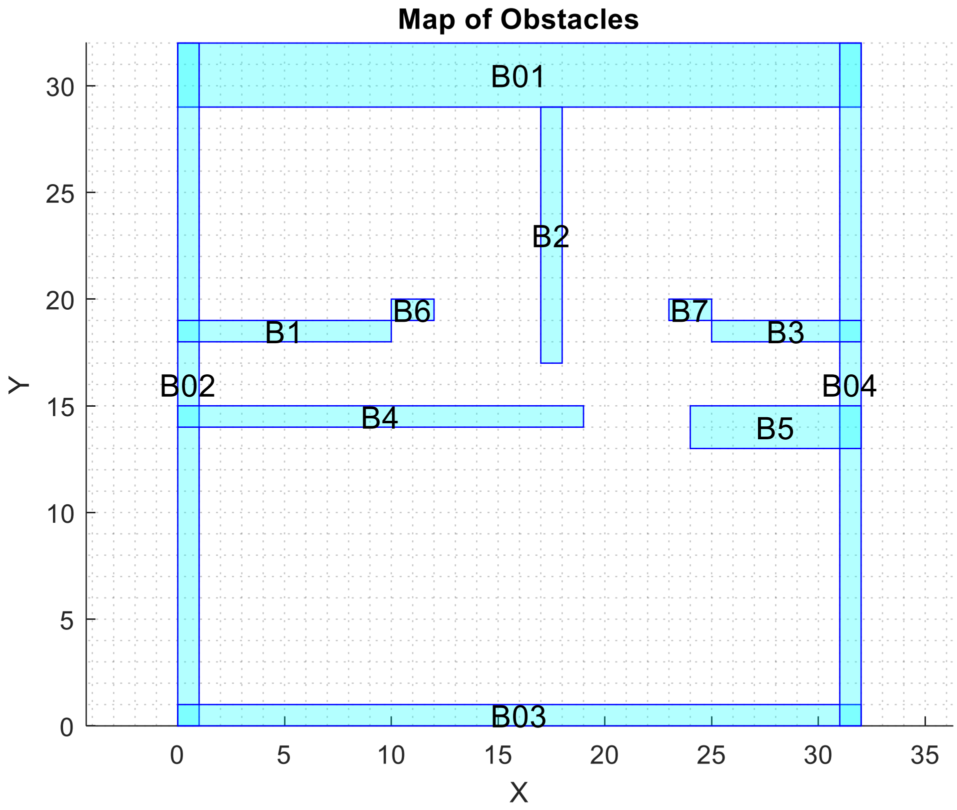
We measured distances between nodes with the Euclidean formula:

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

And set the cost of a move to be that distance plus a small extra cost for rotation. This way, the planner prefers straight lines when possible.

$$Cost = \sqrt{d_x^2 + d_y^2} + 0.1 \cdot \theta$$

The original grid given was 32 × 30, but both parts of the project needed a 32 × 32 grid. To make it fit, we slightly extended obstacle B01 to fill the missing space.



We ran into two main problems. The first was that the grid resolution was too low, if even a tiny part of a cell touched an obstacle in the C-space, the whole cell got marked as blocked. That made paths unnecessarily tight or impossible. We fixed it by doubling the resolution to 64 × 64 reducing the cell size to 0.5 units.

The second was θ resolution. Our robot is an 8 × 1 rectangle rotating about its bottom left corner. The farthest vertex from the pivot is:

$$r = \sqrt{8^2 + 1^2} = \sqrt{65} \approx 8.0622$$

The maximum displacement of that vertex for a single changer in θ is:

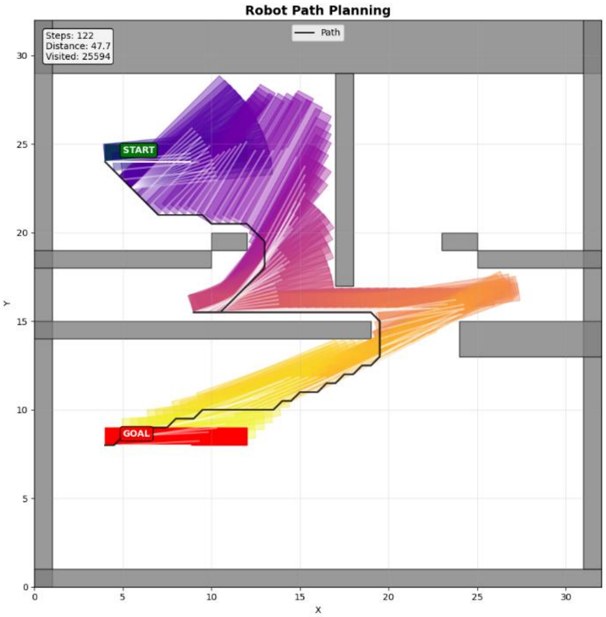
$$\frac{2\pi}{N_{layers}} \cdot r = \frac{16.1244\pi}{N_{layers}}$$

To avoid “skipping” over cells, this displacement must be less than the cell size (0.5 for our grid). That gives:

$$\frac{16.1244\pi}{N_{layers}} \leq 0.5 \rightarrow N_{layers} \geq 101.31$$

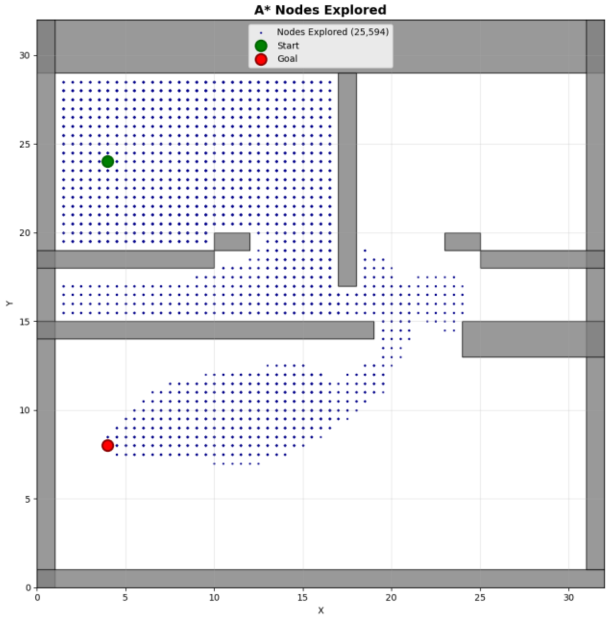
We chose 128 layers to ensure the rotation between consecutive θ slices was small enough.

Putting everything together we get:



As we can see no part of the robot touches the borders and it gets from the start to the goal while rotating. The total number of steps taken was 122 the total distance traveled was 47.7 and the number of nodes needed to get this solution was 25594.

Plotting the nodes explored we get:



As can be seen something something ellipse